MODELAGEM DE APLICAÇÕES DE AUTOMAÇÃO INDUSTRIAL USANDO DIAGRAMAS UML E REUSO DE COMPONENTES

Raimundo Santos Moura*, Luiz Affonso Guedes†

*Departamento de Informática e Estatística
Universidade Federal do Piauí (UFPI)
Teresina – PI – Brasil

† Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte (UFRN)
Natal – RN – Brasil

Emails: rmoura@dca.ufrn.br, affonso@dca.ufrn.br

Abstract— The design engineers need to use high level formalisms to facilitate the development, maintenance, and documentation of the industrial control systems. These formalisms must permit the modeling, formal verification, and code generation to PLC. This paper proposes a high-level methodology for systematize the process of modeling industrial applications of manufacture area. Our proposal use the *State Machine Diagrams* of UML 2.0 and scenarios based in *sequence of events*. An example is presented in details to ease the understanding of our ideas.

Keywords— Modeling, State Machine Diagram, Sequence of events, Industrial applications.

Resumo— Projetistas devem usar formalismos de alto nível para facilitar o desenvolvimento, manutenção e documentação de sistemas de controle industriais. Estes formalismos devem permitir a modelagem, verificação formal e geração de código para PLC. Este artigo propõe uma metodologia de alto nível para sistematizar o processo de modelagem de aplicações industriais da área de manufatura. A proposta utiliza *diagramas de máquina de estados* da UML 2.0 e cenários baseados em *seqüência de eventos*. No final, um exemplo é apresenta em detalhes para facilitar o entendimento de nossas idéias.

Palavras-chave— Modelagem, Diagramas de Máquina de Estados, Seqüência de eventos, Aplicações industriais.

1 Introdução

A automação industrial utiliza a teoria de sistemas para controlar máquinas e processos. Muitos esforços têm sido realizados no desenvolvimento de notações e semânticas usadas para classificar e descrever diferentes tipos de sistemas, sobretudo na área de modelagem. Esses esforços fornecem a infra-estrutura necessária para a solução de alguns problemas reais de engenharia e a construção de sistemas que usam Controladores Lógico Programáveis (do inglês: *Programmable Logic Controller - PLC*), visando principalmente o aumento da produtividade, qualidade e segurança de processos.

Atualmente, a programação dos controladores ainda é realizada por técnicos especificamente qualificados que raramente têm conhecimentos de tecnologias de desenvolvimento de software modernas. Além disso, os controladores são frequentemente reprogramados durante a operação da planta para adaptar novos requisitos. Esses dois pontos justificam a afirmação de que: "não existe uma descrição formal para praticamente nenhum controlador implementado" (Bani Younis and Frey, 2006). Portanto, o uso de metodologias de mais alto nível na programação de controle constitui um grande desafio a ser conquistado e, tais metodologias devem ser simples de usar, acessíveis para análise e semanticamente sólidas.

Na ciência da computação, diversos modelos guiam o processo de desenvolvimento de um *software*. Dentre eles, destacam-se: o **modelo V** (do inglês: *V-Model*) (V-Modell XT, 2004), que foi criado na Alemanha e tem se tornado um modelo clássico para o planejamento e execução de projetos; o **modelo em cascata** (do inglês: *Waterfall model*) (Royce, 1970), usado para desenvolvimento de *software* seqüencial, onde o processo é visto como uma seqüência de fases; e o **modelo espiral** (do inglês: *Spiral model*) (Boehm, 1988), que é um processo de desenvolvimento iterativo e combina elementos dos estágios de projeto e prototipagem de *software*.

Porém, em qualquer modelo de desenvolvimento, o ciclo de vida de uma aplicação pode, resumidamente, ser dividido em: **Modelagem - Validação - Implementação**, conforme apresentado na Figura 1. Pela figura, possíveis iterações no processo modelagem são representadas através do arco "Atualizações". Enquanto que o arco "Reengenharia" representa a área de pesquisa que investiga a geração de modelos a partir de código legado. Nossas pesquisas concentram-se no processo de modelagem direta, ou seja, na modelagem de aplicações a partir de requisitos especificados pelos usuários.

Existem na literatura diversas abordagens que apresentam metodologias, linguagens e padrões para a modelagem de aplicações industriais, em

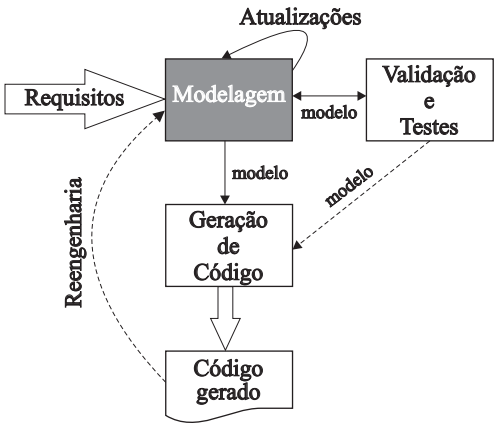

Figura 1: Ciclo de vida resumido de uma aplicação

especial, sistemas a eventos discretos (Cassandras and Lafortune, 1999). Os *Statecharts* (Harel, 1987) representam a principal abordagem e consistem na evolução dos autômatos de estados finitos. Com a definição da *Unified Modeling Language (UML)*, proposta pelo consórcio *Object Management Group (OMG)*, os *Statecharts* foram incorporados para descrever aspectos dinâmicos do comportamento do sistema. A UML descreve os *Statecharts* com uma semântica ligeiramente diferente da semântica original proposta por Harel (Harel and Politi, 1998). Na versão mais recente (UML 2.0), os *Statechars* passaram a ser chamados de *State Machine Diagram*. Porém, eles apresentam poucas alterações com relação aos *Statecharts* das versões anteriores.

O uso dos *Statecharts* tem sido bastante explorado, por exemplo, os trabalhos (Bani Younis and Frey, 2006), (Bates et al., 1998), (Machado et al., 2001), (Huang et al., 2004), (Sacha, 2005), (Hu and Shatz, 2006) e (Secchi et al., 2007) apresentam algumas contribuições nesta área. Entretanto, as técnicas de modelagem apresentadas possuem forte dependência da experiência do projetista e, de nosso conhecimento, nenhuma delas descreve um método sistemático para a criação do modelo.

Com intenção de preencher o *gap* existente entre requisitos informais e o processo de modelagem, este trabalho propõe uma metodologia de alto-nível para modelagem de aplicações industriais, especialmente aplicações da área de manufatura.

O restante deste artigo está organizado da seguinte maneira: a seção 2 apresenta uma visão geral dos *diagramas de máquina de estados* e discute os principais aspectos para o seu uso na modelagem de sistemas de automação industrial. A seção 3 apresenta em detalhes a técnica de modelagem proposta nesta contribuição. Em seguida, na seção 4 descreve-se um exemplo completo de uma aplicação industrial usando a metodologia proposta. Finalmente, a seção 5 conclui este artigo e destaca alguns trabalhos futuros.

2 UML 2.0: Diagramas de Máquina de Estados

Apesar da UML possuir diversos diagramas que podem ser usados para mostrar algumas partes do modelo de um sistema, é possível utilizá-la como um "rascunho"e ter seus modelos contendo apenas alguns diagramas estruturais e comportamentais. Para sistemas de manufatura, em geral, os diagramas de máquina de estados e os diagramas de seqüência são os mais comumente utilizados.

Os diagramas de máquina de estados são úteis na modelagem de componentes que podem estar em diversos estados e que possuam eventos causando mudanças de estado. Um exemplo clássico modela o comportamento de uma lâmpada que pode estar nos estados "LIGADO" ou "APAGADO". A mudança do estado "LIGADO" para "APAGADO" é provocado pelo disparo de um evento, tal como o pressionamento de um botão (interruptor).

A UML 2.0 define uma *máquina de estado comportamental*, que pode mostrar estados, transições e comportamentos (dentro dos estados e junto das transições). Entretanto, existe outro tipo de máquina de estados - *máquina de estados de protocolo* - que é útil para modelagem de protocolos tais como protocolos de comunicação de redes. Nesta contribuição apenas as máquinas de estados comportamentais serão utilizadas.

Podemos citar as seguintes vantagens do uso dos *Statecharts*, e por conseguinte, dos *diagramas de máquina de estados* para a modelagem de sistemas orientados a eventos:

- são um formalismo gráfico, simples e intuitivo;
- permitem níveis de estados (*clustering, unclustering e refinamentos*) através de **decomposição-OR**: estando em um estado o sistema deve estar em apenas um de seus subestados. *Clustering e Unclustering* permitem que o projetista explore a idéia de "zoom-in" e "zoom-out" para esconder/detalhar níveis de abstração da modelagem do sistema;
- possuem ortogonalidade de estados (*independência e concorrência*) através de **decomposição-AND**. Isso significa que: estando em um estado o sistema deve estar em todos os seus subestados paralelos;
- permitem restrições temporais através do uso de temporizadores implícitos. Formalmente, a expressão **timeout(e, t)** representa o evento que deve ser disparado quando o número especificado **t** de unidades de tempo decorrer após a ocorrência do evento **e**;

- associam **ações e condições** aos eventos de uma transição de estados e **atividades** que podem ser disparadas na entrada, saída ou durante um estado. Formalmente, o evento de uma transição é representado pela expressão $e[c]/a$, onde o evento **e** deve ocorrer apenas se a condição opcional **c** for verdadeira. Neste caso, a ação **a** deve ser disparada automaticamente;

- Fazem parte da *Systems Modeling Language (SysML)* (SysML, 2006). SysML é uma linguagem de modelagem de domínio específico para aplicações de engenharia de sistemas, que suporta a especificação, análise e projeto de uma larga faixa de sistemas e metasistemas. Ela é recomendada pelo consórcio OMG para a modelagem de sistemas que podem incluir hardware e software.

Deste modo, acredita-se que os *diagramas de máquina de estados* sejam um formalismo bastante poderoso e contemplam as necessidades da especificação da aplicações industriais.

3 Metodologia Proposta

A metodologia proposta utiliza *diagramas de máquina de estados* em conjunto com cenários definidos através de *seqüência de eventos* e consiste na aplicação das etapas descritas a seguir:

1. Modelar os elementos básicos da aplicação (por exemplo: cilindros, válvulas e sensores) ou usar modelos definidos em um repositório de componentes;

2. Decompor os estados em possíveis subestados, como por exemplo: um cilindro pode estar "RECUADO"ou "AVANÇADO", uma válvula pode estar "FECHADA" ou "ABERTA", entre outros;

3. Representar todos os componentes/etapas da planta como estados que funcionam em paralelo, herdando as características dos elementos básicos, quando possível;

4. Criar cenários baseados em seqüência de eventos para refinar cada estado e incluir características especiais do funcionamento da aplicação.

As etapas 1 e 2 consistem na modelagem e refinamentos dos elementos básicos que compõem a aplicação. Tais etapas formam um ciclo e podem ser executadas através de várias iterações. Em cada iteração se trabalha com componentes mais complexos. Além disso, eles podem ser agrupados em um repositório e são exemplificados na subseção 3.1. A etapa 3 indica que todos os componentes da aplicação devem ser executados concorrentemente e a etapa 4 é o passo responsável pela interconexão dos componentes, ou seja, ela representa as possíveis interações entre os componentes envolvidos na aplicação.

3.1 Sistemas de manufatura: componentes básicos

De um modo geral, uma aplicação pode estar em dois possíveis estados: *"ForaOperacao"* e *"EmOperacao"*. A Figura 2 mostra o comportamento básico de uma aplicação. Inicialmente, ela se encontra no estado *"ForaOperacao"*, indicado através da seta pequena. Tal estado é chamado de *estado default*. De acordo com a figura, a aplicação entra em funcionamento (estado: *"EmOperacao"*) através do evento "start" e finaliza (estado: *"ForaOperacao"*) através do evento "end".

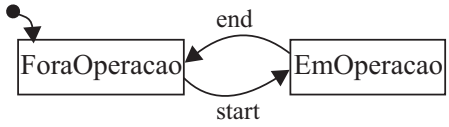

Figura 2: Aplicação genérica: modelo básico

Para sistemas de automação da área de manufatura alguns exemplos de componentes são: cilindros, válvulas e sensores. A modelagem do comportamento básico destes elementos é apresentada na Figura 3. Pela figura, um cilindro pode estar nos estados "RECUADO" ou "AVANCADO", sendo RECUADO o estado inicial que é indicado pela pequena seta.

A transição "ac" permite o cilindro mudar do estado "RECUADO" para "AVANCADO", enquanto que a transição "rc" permite mudar do estado "AVANCADO" para "RECUADO". O funcionamento de uma válvula e de um sensor segue o mesmo padrão apresentado para o cilindro. Os rótulos *ac*, *rc*, *vOn*, $vOff$, *sOn* e $sOff$ representam os eventos para avançar cilindro, recuar cilindro, abrir válvula, fechar válvula, sensor detecta algo e sensor sem detectar algo, respectivamente. Assim, de modo geral esses componentes podem ser modelados através de dois estados: *On e Off*.

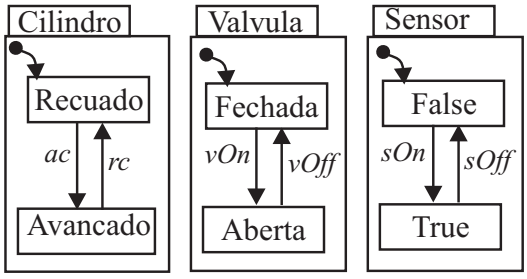

Figura 3: Componentes: modelo básico

Basicamente, os cilindros usados nestas aplicações podem ser de dois tipos:

- cilindros de ação simples - onde o avanço é controlado por válvula e o retorno é realizado por molas;

- cilindros de ação dupla - onde o avanço e o retorno são controlados por válvulas.

Ambos os cilindros podem ter seus avanços e recuos monitorados por sensores de início e final de curso (ver Figura 4). Aqui, as letras a e r utilizadas nos eventos das válvulas e sensores representam *Avanço* e *Recuo*, respectivamente. Por exemplo, *vaOn* significa válvula de avanço aberta e $srOff$ significa sensor de recuo total não acionado. Os rótulos citados nesta seção serão usados ao longo do trabalho. Além disso, sempre que existir mais de um componente do mesmo tipo na aplicação, uma numeração seqüencial será utilizada. Desta forma, ac_i significa avançar cilindro i, para i=1, 2, ...; vr_iOff significa válvula de recuo i fechada, para i=1, 2, ...; e assim sucessivamente.

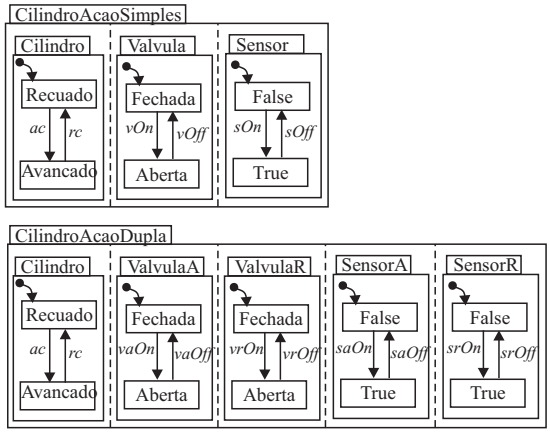

Figura 4: Cilindros: modelo básico

Para aplicações que precisam manter um cilindro avançado por um determinado tempo ou que precisam verificar se o cilindro alcançou o avanço total em um tempo especificado, os sensores podem ser remodelados para permitir tais características (ver Figura 5). Em *SensorComTimer*, quando a transição *sOn* (sensor de fim de curso ativo) é executada, um temporizador t é iniciado através de uma atividade na entrada do estado "TRUE". Após t segundos a transição $tm(t)/action$ ocorre e o estado ativo é alterado para "WAIT". O retorno ao estado "FALSE" ocorre através da transição $sOff$ (sensor de fim de curso desligado). Em *SensorComExcecao*, quando a transição $timerT$ é executada, um temporizador t é iniciado através de uma atividade na entrada do estado "WAIT". A partir deste estado, dois caminhos são possíveis: i) o sensor de fim de curso *sOn* é percebido, alterando o estado ativo para "TRUE"e o retorno ao estado "FALSE" ocorre através da transição $sOff$; e ii) após t segundos a transição $tm(t)/action$ ocorre, o estado ativo é alterado para "ERROR"e o retorno ao estado "FALSE" ocorre através da transição $recuperarError$.

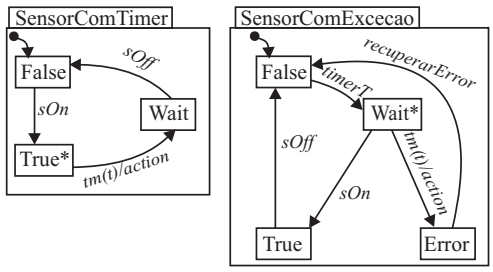

Figura 5: Sensores temporizados: modelo básico

3.2 Cilindro de Ação Simples: funcionamento

Esta subseção exemplifica o funcionamento de um cilindro de ação simples, cujo avanço é executado através do pressionamento de um botão. O comportamento do mesmo é obtido através da definição de cenários baseados em seqüência de eventos. Esses passos correspondem às etapas 3 e 4 do processo de modelagem proposto.

Dado o modelo básico de uma aplicação genérica, o estado *"EmOperacao"* deve conter um componente com as características de um cilindro de ação simples, ou seja, *EmOperacao* **contains** *cilindro1;* e o *cilindro1* **is** *CilindroAcaoSimples;*.

O cenário que representa o funcionamento correto da aplicação é definido pelo projetista através de uma seqüência de eventos baseado no diagrama de seqüência da UML (ver Figura 6). Observa-se que os eventos internos (setas tracejadas) representam as ações que devem ser executadas no modelo para facilitar a simulação do comportamento da aplicação, por exemplo, os eventos "ac1"e "rc1"atuam apenas no modelo. Já os eventos "v1On", "v1Off"e "end" atuam tanto no modelo quanto na planta. Eventos externos ocorrem na planta e são percebidos pelo modelo através de sensores (por exemplo: um botão ser pressionado, um sensor detectar presença/ausência de alguma coisa, etc.) ou eles ocorrem no modelo e atuam na planta (por exemplo: abertura/fechamento de uma válvula, liberação de botão para pressionamento, etc.).

De acordo com o cenário especificado para uma aplicação, o modelo do sistema sofre acréscimos para incluir as características definidas. Os acréscimos são apresentados a seguir:

- a transição "*start*" é acrescida da ação "$v1On$", tornando-se "$start/v1On$". Isso significa que: quando o botão "*start*" for pressionado o sistema deve entrar em funcionamento e fazer a abertura da válvula1, provocando o avanço do cilindro1;

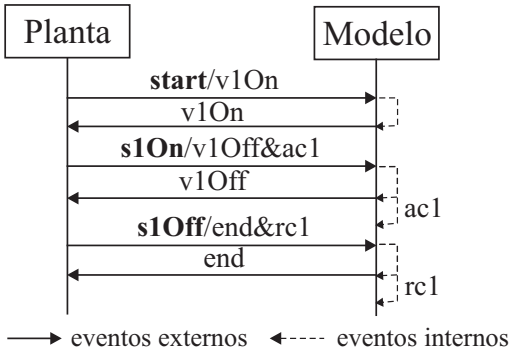

Figura 6: Cilindro de ação simples: eventos

- a transição "$s1On$", acrescida da ação correspondente torna-se "$s1On/ac1\&v1Off$". Significa que: quando o evento "$s1On$" ocorrer (isto é, o cilindro1 atingir o avanço total) o sistema deve alterar o estado do cilindro1 para "AVANCADO" e a válvula1 deve ser fechada;
- a transição "$s1Off$" torna-se "$s1Off/rc1\&end$". Isso significa que: quando o cilindro1 retornar, o sistema deve alterar o estado do cilindro1 para "RECUADO" e finalizar a execução da aplicação.

O modelo completo do cilindro em questão é apresentado na Figura 7. Nota-se que o comportamento do Cilindro1 e da Valvula1 não foi alterado. Isso ocorre porque a nossa metodologia tenta manter uma relação um-para-um entre a planta e o modelo. Dessa forma, alguns componentes da planta são representados no modelo apenas para facilitar o entendimento da aplicação, que pode ser observado através de simulações e também, para manter a fidelidade com o sistema real.

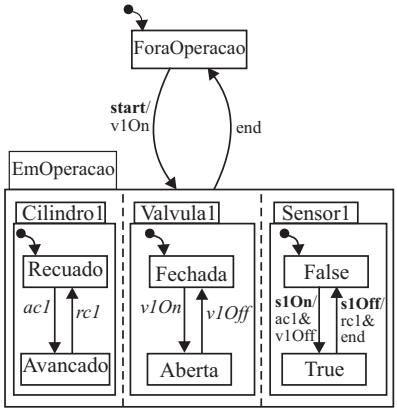

Figura 7: Cilindro de ação simples: modelo final

É importante ressaltar que as alterações indicadas podem ser realizadas sistematicamente no modelo do sistema. Portanto, o que um projetista deve fazer para criar uma aplicação usando o repositório de componentes pré-definidos é, basicamente: i) especificar os componentes que estendem as características dos elementos básicos; e ii) especificar os cenários que determinam o funcionamento desejado da aplicação.

A nossa metodologia possibilita ainda a modelagem de sistemas de produção flexíveis, através da mudança de cenários. Assim, a partir do comportamento básico dos componentes do sistema, diferentes cenários podem gerar modelos alternativos do *software* controlador e, consequentemente, podem provocar execuções diferentes da planta. Sistemas de produção flexíveis estão fora do escopo deste artigo.

A validação do funcionamento da aplicação e, portanto, do modelo gerado pode ser feito através de simulações. Tal processo também está fora do escopo desta contribuição. Porém, em (Moura and Guedes, 2007), apresenta-se uma simulação de aplicação industrial usando o ambiente de execução SCXML para a modelagem do controlador e tecnologias Java 2D para simulação da planta.

4 Estudo de Caso

Nesta seção, descreve-se um exemplo completo de aplicação industrial da área de manufatura, que utiliza a metodologia proposta. Este exemplo foi retirado do livro "Automação e Controle Discreto" de Paulo Silveira (da Silveira and dos Santos, 1998).

4.1 Equipamento para estampar peças

4.1.1 Definição

O equipamento para estampar peças (ver Figura 8) é formado por um dispositivo de carregamento de peças (não representado na figura), **um cilindro alimentador - cilindro1**, um **cilindro estampador - cilindro2** e um **cilindro extrator - cilindro3**. Todos os cilindros são de ação simples com retorno por mola e têm seus avanços controlados por válvulas. Cada cilindro possui um sensor de fim de curso, que indica o avanço total do mesmo. A expulsão da peça é realizada por um sopro de ar comprimido, obtido a partir do acionamento de uma quarta válvula (V4) que é efetivamente monitorada pela atuação de um fotosensor.

A configuração inicial do sistema em funcionamento prevê que todos os cilindros estejam recuados, as válvulas fechadas e o fotosensor sem detectar peças. O sistema entra em operação através do acionamento do botão "start" e, é desligado automaticamente, quando o fotosensor detecta a presença de uma peça. O cenário de funcionamento da aplicação é o seguinte:

"O cilindro alimentador empurra

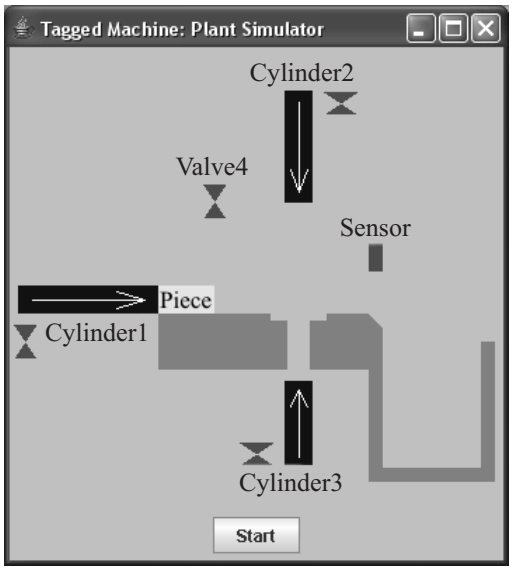

Figura 8: Etiquetador: simulação

uma peça até o molde; o cilindro estampador prensa um estampo sobre a peça por dois segundos; o cilindro extrator em conjunto com a válvula V4 realiza a retirada da peça pronta, empurrando-a para o depósito".

4.1.2 Modelagem

Como este exemplo utiliza somente componentes discutidos na subseção 3.1, considera-se a existência de um repositório de componentes. Assim, a modelagem se inicia a partir da etapa 3, com a representação de todos os componentes da planta no estado *"EmOperacao"* de uma aplicação genérica (ver Figura 9), que funcionam de maneira paralela.

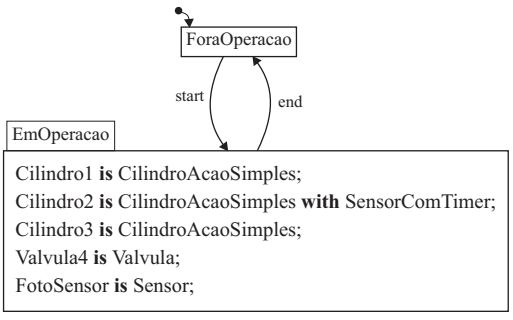

Figura 9: Etiquetador: modelo básico

A Figura 10 mostra a seqüência de eventos disparados para o cenário desejado do exemplo em questão. Isso corresponde à etapa 4 da metodologia proposta. As linhas tracejadas representam eventos internos ao modelo e descrevem as características especiais do funcionamento da aplicação. Estes eventos permitem refinar os estados envolvidos, incluindo tais características. As linhas contínuas representam eventos externos ao modelo, ou seja, eventos que ocorrem na planta e são percebidos pelo modelo através dos sensores ou eventos gerados no modelo e que atuam na planta através dos atuadores.

O modelo final gerado consiste no funcionamento, em paralelo, dos três cilindros, das quatro válvulas e dos sensores (ver Figura 11).

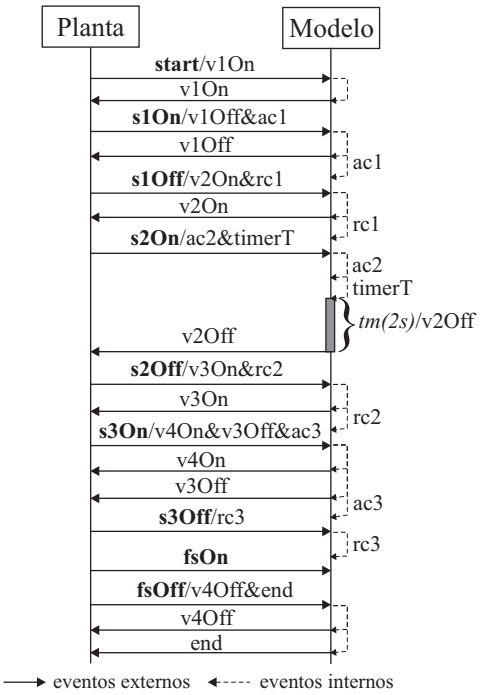

Figura 10: Etiquetador: eventos

5 Conclusão

Este trabalho apresentou uma metodologia para sistematizar o processo de modelagem de aplicações industriais, em especial, aplicações da área de manufatura. A proposta utiliza os *diagramas de máquinas de estados* da UML 2.0 em conjunto com cenários baseados nos *seqüência de eventos*. A modelagem é baseada em quatro etapas que podem ser executadas através de várias iterações. De modo geral, ela representa uma *abordagem de desenvolvimento bottom-up*, permite reuso de componentes e mantém uma relação um-para-um entre a planta e o modelo, ou seja, uma fidelidade do modelo com o sistema real. Um exemplo típico de aplicação da área de manufatura foi descrito em detalhes para facilitar o entendimento da proposta.

Atualmente, está sendo desenvolvendo na linguagem Java uma ferramenta computacional para criar e simular os modelos gerados através da metodologia proposta. Em paralelo, pretende-se discutir reconfiguração de sistemas através de troca

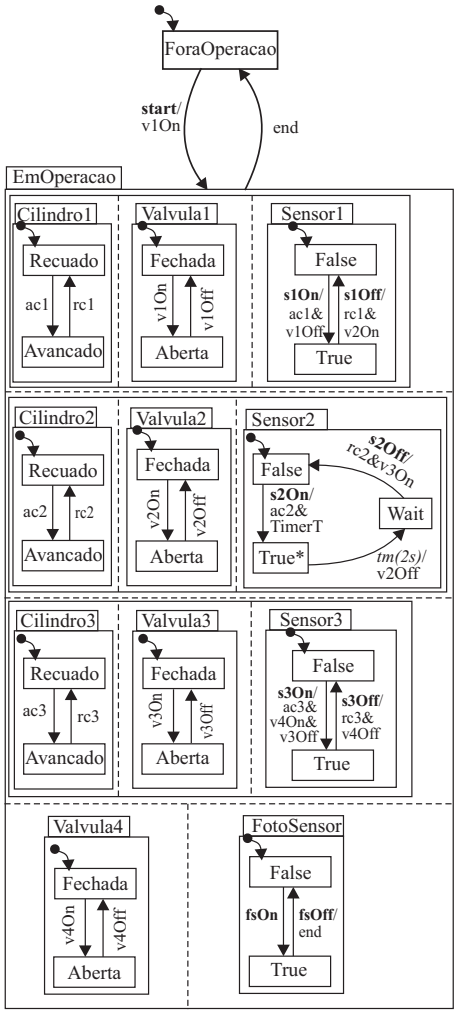

Figura 11: Etiquetador: modelo final

de cenários, além de aplicar a proposta em outras áreas de aplicações. Finalmente, acredita-se que alguns padrões de componentes poderão ser especializados e, consequentemente, alguns conectores, tais como **is, with, contains**, poderão ser usados para formalizar uma linguagem de descrição textual do formalismo.

Referências

Bani Younis, M. and Frey, G. (2006). Uml-based approach for the re-engineering of plc programs, *32nd Annual Conference of the IEEE Industrial Electronics Society (IECON'06)*, pp. 3691–3696.

Bates, I. D., Chester, E. G. and Kinniment, D. J. (1998). A case study in the automatic programming of a plc based control system using statemate statecharts, *UKACC International Conference on Control. IEEE* **1**: 832–837.

Boehm, B. W. (1988). A spiral model of software development and enhancement.

Cassandras, C. G. and Lafortune, S. (1999). *Introduction to Discrete Event Systems*, Kluwer Academic Publishers.

da Silveira, P. R. and dos Santos, W. E. (1998). *Automação e Controle Discreto*, São Paulo - SP: Érica.

Harel, D. (1987). Statecharts: A visual formalism for complex systems, *Science of Computer Programming* **8**(3): 231–274.

Harel, D. and Politi, M. (1998). *Modeling Reactive Systems with Statecharts: the Statemate Approach*, McGraw Hill.

Hu, Z. and Shatz, S. M. (2006). Explicit modeling of semantics associated with composite states in uml statecharts, *Automated Software Engg.* **13**(4): 423–467.

Huang, Y., Jeng, M. and Hsu, C. (2004). Modeling a discrete event system using statecharts, *International Conference on Networking, Sensing & Control. IEEE* **2**: 1093–1098.

Machado, J. M., Louni, F., Faure, J.-M., Lesage, J.-J., da Silva, J. C. L. F. and Roussel, J.-M. (2001). Modelling and implementing the control of automated production systems using statecharts and plc programming languages, *European Control Conference (ECC2001)*, Porto, Portugal.

Moura, R. S. and Guedes, L. A. (2007). Simulation of industrial applications using the execution environment scxml, *5th IEEE International Conference on Industrial Informatics (INDIN 2007)*, Vienna, Austria.

Royce, W. W. (1970). Managing the development of large software systems, *Proc. of IEEE WESCON*, pp. 1–9.

Sacha, K. (2005). Automatic code generation for plc controllers., *SAFECOMP*, pp. 303–316.

Secchi, C., Bonfe, M. and Fantuzzi, C. (2007). On the use of uml for modeling mechatronic systems, *IEEE Transactions on Automation Science and Engineering*, Vol. 4, pp. 105–113.

SysML (2006). Systems modeling language (sysml), http://www.sysml.org/.

V-Modell XT (2004). Part 1: Fundamentals of the v-modell, http://www.v-modell-xt.de/.